FUNCIONES GLOBALES EN JAVASCRIPT

Son **funciones** que no pertenecen a ningún objeto, sino que forman parte del propio núcleo de **JavaScript**.

escape() Codifica los caracteres no alfanuméricos de una cadena para que pueda formar parte de una URL. Por ejemplo, para especificar un valor de parámetro que contiene espacios:

```
// url no válida (el parámetro contiene espacios)
url = "cgi-bin/prog.pl?cadena=esta cadena";

// ahora con 'escape':
url = "cgi-bin/prog.pl?cadena=";
url += escape ("esta cadena");
// Resultado: cgi-bin/prog.pl?cadena=esta%20cadena
```

No codifica los caracteres especiales: *** @ - _ + . /**

eval() Evalúa una expresión JavaScript (como lo haría el analizador) sin necesidad de referenciar un objeto determinado ni una variable declarada.

```
sentencias = new Array(2);
sentencias[0]="document.write ('expresion 1')";
sentencias[1]="suma = 2+2";
if (condicion) eval (sentencias[0]);
else eval (sentencias[1]);
```

isNaN() Devuelve *true* si el valor pasado como argumento no es un número

parseFloat() Evalúa una cadena de texto que expresa un número en notación científica y devuelve el número en coma flotante correspondiente ó NaN en caso de no encontrar una expresión válida. Ejemplo:

```
valor = parseFloat ("1.4142e-10");
valor1 = parseFloat (".0012");
valor2 = parseFloat ("a12");   // devuelve NaN
```

Si la cadena no representa un número en coma flotante válido, la función devuelve NaN:

```
valor3 = parseFloat ("12.8abc"); // valor3 = 12.8
valor4 = parseFloat ("abc12.8"); // valor4 = NaN
```

parseInt() Evalúa una cadena de texto y devuelve el entero representado por dicha cadena. Se puede especificar la base en la que se expresa el número:

```
entero1 = parseInt ("FF", 16);   // entero1 = 255
entero2 = parseInt ("1010", 2); // entero2 = 10
```

Si no se especifica la base, se sigue el siguiente criterio:
- Los números que comienzan por 0 se considerarán en base octal.
- Los números que comienzan por 0x se consideran hexadecimales.
- El resto de los números se consideran enteros en base 10.

Si la cadena no representa un número válido, la función devuelve NaN. Ejemplos:

```
entero3 = parseInt ("123abcd"); // entero3 = 123
entero4 = parseInt ("3.14");      // entero4 = 3
entero5 = parseInt ("abc123");  // entero5 = NaN
```

unescape() Operación contraria a *escape()*. Decodifica los caracteres especiales de la URL.

MÉTODOS COMUNES A TODOS LOS OBJETOS

Son **Métodos** que están disponibles en todos los objetos.

toString() Devuelve una cadena que representa al objeto.

valueOf() Devuelve el valor primitivo del Objeto.

EL OBJETO DATE (FECHA) EN JAVASCRIPT

Este **objeto** permite trabajar con fechas y horas en los programas. Por ejemplo, la sentencia:

```
hoy = new Date()
```

Crea un nuevo objeto **Date** llamado 'hoy' y le asigna la fecha y la hora actuales.
Para trabajar con este objeto se suelen utilizar estos **métodos**:

getDate()	*Devuelve el día del mes*
getDay()	*Día de la semana*
getHours()	*Devuelve la hora*
getMinutes()	*Devuelve los minutos*
getMonth()	*Devuelve el mes*
getSeconds()	*Devuelve los segundos*
getTime()	*Devuelve un valor numérico asociado al instante para la fecha especificada*
getTimezoneOffset()	*Devuelve el desfase horario (en minutos) del sistema local con respecto al meridiano cero*
getYear()	*Devuelve el año. El formato utiliza dos dígitos para años comprendidos entre 1900 y 1999. Para el resto utiliza cuatro dígitos.*
parse()	*Devuelve el número de milisegundos desde las 00 horas del 1 de enero de 1970 hasta el instante representado por la fecha que recibe como parámetro:* *milisegundos = Date.parse("Mon, 25 Dec 1995 13:30:00 GMT+0430")* *milisegundos2 = Date.parse ("Mon, 25 Dec 1995");* *El formato de la fecha es el indicado en el ejemplo. Si no se incluye la zona horaria, se utiliza la zona horaria del sistema.*
setDate()	*Asigna el día del mes a un objeto Date*
setHours()	*Asigna la hora*
setMinutes()	*Asigna los minutos*
setMonth()	*Asigna el mes*
setSeconds ()	*Asigna los segundos*
setTime()	*Asigna el valor del objeto Date*
setYear()	*Asigna el año*
toGMTString()	*Convierte la fecha a cadena de texto, utilizando como referencia la hora del meridiano cero (GMT).*
toLocaleString()	*Convierte la fecha a cadena de texto, utilizando como referencia la configuración horaria del sistema.*
UTC	*Devuelve el número de milisegundos desde las 00 horas del 1 de enero de 1970 hasta la fecha almacenada por el objeto Date*

El objeto Date sólo representa un cierto instante de tiempo (no se actualiza automáticamente). Para consultar el instante actual se puede crear un nuevo objeto Date utilizando el operador 'new'. Para crear un objeto Date con una fecha arbitraria se puede utilizar el constructor:

```
aquel_dia = new Date(2000, 0, 1, 2, 30, 30)
```

En el ejemplo anterior se crea un objeto Date que representa el instante: 2 horas, 30 minutos, 30 segundos del 1 de Enero (enero = 0, febrero=1, …) de 2000.

EL OBJETO ARRAY EN JAVASCRIPT

La creación de un **array** se lleva a cabo con la sentencia:

```
mi_array = new Array();
```

Se pueden asignar a un array valores de cualquier tipo (incluyendo objetos).

```
mi_array[0] = 1;
mi_array[1] = "Cadena de texto";
mi_array[2] = Date();
```

Tanto el tamaño (longitud) del array como el tamaño de las celdas individuales se asigna de forma dinámica (automática) a medida que se asignan elementos. Hay que tener en cuenta que el tamaño de los arrays en **JavaScript** puede crecer pero no decrecer, es decir, para el siguiente trozo de código:

```
var array_grande = Array();
array_grande[1000]=1;
```

Se construye un array de 1001 posiciones (aunque realmente sólo se utiliza una de ellas). Se pueden utilizar cadenas de texto como identificadores de índices (**arrays asociativos**) en la versión de Navigator 3 y posteriores. El constructor permite crear arrays de una determinada longitud inicial:

```
otro_array = new Array (10);
```

E inicializar valores a partir de la declaración:

```
este_array = new Array ("uno", "dos", "tres", "cuatro");
```

Propiedades del objeto Array:

length	Número de elementos del array

Métodos:

concat()	Une dos arrays y devuelve como resultado un array con la unión
join()	Devuelve una cadena de texto que contiene la unión de los elementos del array
pop()	Borra y devuelve el último elemento de un array
push()	Añade un elemento a un array y devuelve ese elemento
reverse()	Refleja el contenido de un array (el primer elemento pasa a ser el último)
shift()	Borra y devuelve el primer elmento de un array

slice()	Extrae una sección de un array y la devuelve como un nuevo array:
	`datos = new Array ()`
	`datos[1]="uno";`
	`datos[2]="dos";`
	`datos[3]="tres";`
	`dosprimeros = datos.slice(1,2);`
	`// dosprimeros[0]="uno"`
	`// dosprimeros[1]="dos"`

splice()	Añade y/o elimina elementos de un array

sort()	Ordena los elementos de un array. Por defecto utiliza ordenación lexicográfica (alfabética). Se puede definir una función externa que implemente el criterio de ordenación:
	`function compareNumbers(a, b) {`
	`  return a - b}`
	`numberArray.sort(compareNumbers)`

toString()	Devuelve una cadena que representa al array
unshift()	Añade uno o más elementos al comienzo de un array y devuelve el número actualizado de elementos

El objeto Array no existe en JavaScript 1.0 (Netscape 2.x y MIE 3.x). Sin embargo se puede utilizar una aproximación a través de funciones y arrays normales para conseguir resultados similares.

EL OBJETO MATH EN JAVASCRIPT

Es un **objeto** de nivel superior, predefinido en **JavaScript**. No necesita **constructor**.

Propiedades del objeto Math:

E	Constante de Euler. Base de los logaritmos neperianos (2.718…).
LN10	Logaritmo neperiano de 10. Aprox. 2.302...
LN2	Logaritmo neperiano de 2. Aprox. 0.693...
LOG10E	Logaritmo en base 10 de E. Aprox. 0.434...
LOG2E	Logaritmo en base 2 de E. Aprox. 1.442...
PI	Constante PI. Aprox. 3.14159…
SQRT1_2	Raíz de 1/2. Aprox. 0.707...
SQRT2	Raíz de 2. Aprox. 1.414...

Métodos del objeto Math:

abs()	Valor absoluto de un número
acos()	Arco coseno de un número (en radianes)
asin()	Arco seno de un número (en radianes)
atan()	Arco tangente de un número (en radianes)
atan2(x,y)	Angulo que forma el punto (x,y) con respecto al eje X
ceil()	Entero inmediatamente superior del número
cos()	Coseno de un ángulo expresado en radianes
exp()	Función exponencial
floor()	Entero inmediatamente inferior al número dado
log()	Logaritmo neperiano de un número
max(x,y)	Devuelve el mayor de los dos números
min(x,y)	Devuelve el menor de los dos números
pow(a,b)	Devuelve 'a' elevado a 'b'
random()	Devuelve un número pseudoaleatorio comprendido entre 0 y 1
round()	Redondea un número al entero más cercano
sin ()	Seno de un ángulo expresado en radianes
sqrt()	Raíz cuadrada de un número
tan()	Tangente de un ángulo expresado en radianes

El acceso a las propiedades y métodos de Math se realiza como si de otro objeto se tratase (sin instanciar nuevos objetos Math):

```
resultado = 2 * Math.PI;
seno = Math.sin(ángulo);
```

EL OBJETO STRING EN JAVASCRIPT

Es un objeto predefinido de **JavaScript** para manejo de cadenas de caracteres.

Propiedades del objeto String

length Longitud de la cadena de caracteres asociada al objeto

Se crea con el **constructor String**, pero cualquier variable que contiene una cadena de texto puede ser tratada como String (JavaScript se encarga de hacer las conversiones necesarias):

```
cadena1 = new String("Un String");
longitud = cadena1.length;

cadena2 = "Una cadena de caracteres";
longitud = cadena2.length; // se convierte temporalmente a String para utilizar la propiedad length
```

Métodos del objeto String

anchor(Name) Crea un anclaje HTML:

```
referencia = new String ("Zona inferior");
document.write(referencia.anchor ("inferior"));
```

Sería equivalente a: *Zona inferior*

big() Devuelve una cadena que representa el texto en formato BIG de HTML

bold() Devuelve una cadena que representa el texto en negrita (...) en HTML

charAt(index) Devuelve el caracter correspondiente a la posición indicada (teniendo en cuenta que las cadenas comienzan con índice cero)

charCodeAt(index) Devuelve el código (codificación ISO-Latin-1) correspondiente al carácter situado en la posición indicada (las cadenas comienzan con índice cero)

concat Concatena dos cadenas y devuelve como resultado esa unión

fixed() Devuelve una cadena con el texto en formato TT (ancho fijo) de HTML

fontcolor(color) Devuelve una cadena con el texto del color especificado

fontsize(tamaño) Devuelve una cadena con el texto del tamaño especificado

indexOf(texto, desde) Devuelve el índice de la primera aparición del valor especificado dentro de la cadena representada por el objeto:

```
cadena = "uno dos tres cuatro";
posicionDeDos = cadena.indexOf("dos"); // posicionDeDos = 4
```

italics() Devuelve una cadena que representa el texto en formato <I>...</I> (cursiva) de HTML

lastIndexOf(texto, desde) Devuelve el índice de la última aparición del valor especificado dentro de la cadena representada por el objeto:

```
cadena = "uno dos tres cuatro dos";
posicionDeDos = cadena.lastIndexOf("dos"); // posicionDeDos = 20
```

link(href) Construye una cadena que se corresponde con la marca de enlace (<a>) de HTML:

```
enlace = new String("Epsilon Eridani");
marcaEnlace = enlace.link ("http://www.epsilon-eridani.com");
// marcaEnlace = <a href='http://...'>Epsilon Eridani</a>
```

slice(inicio,fin) Extrae un trozo de una cadena: (**inicio** comienza en 0, pero **fin** comienza en 1)

```
cadena = "dos trozos";
trozo = cadena.slice(4,9); // trozo='trozo'
```

Si no se especifica **fin**, se considera hasta el final de la cadena.

Si **fin** es negativo, indica la posición comenzando por la parte derecha de la cadena original:

```
trozo = cadena.slice(4,-1); // trozo='trozo'
```

small() Devuelve una cadena que se corresponde con el formato SMALL de HTML

split(separador) Divide una cadena en trozos y los devuelve en un array. Se pasa como parámetro la cadena utilizada como separador:

```
cadena = "uno dos tres cuatro";
trozos = cadena.split(" "); // el separador es el espacio
document.write(trozos[0]); // trozos[0] = 'uno'
```

Se puede especificar un segundo parámetro que indica el número máximo de trozos:

```
trozos = cadena.split(" ",2); // se crea un array de 2 elementos
```

strike() Devuelve una cadena que representa el texto en formato STRIKE (Tachada) de HTML

sub() Devuelve una cadena con el formato SUB (subíndice) de HTML

substr(inicio, longitud) Devuelve una subcadena a partir de los parámetros **inicio** y **longitud**:

```
cadena = "uno dos tres cuatro";
subcadena = cadena.substr(4, 3); // subcadena = 'dos'
```

Si no se especifica la **longitud**, se considera hasta el final de la cadena.

substring(inicio, fin) Parecido a **slice()**. Devuelve una subcadena que está comprendida entre dos índices de la cadena actual (**inicio** comienza en 0, pero **fin** comienza en 1):

```
cadena = "uno dos tres cuatro";
subcadena = cadena.substring(4, 7); // subcadena = 'dos'
```

Se pueden invertir los parámetros (**fin**, **inicio**) pero entonces (**inicio** comienza en 1 y **fin** comienza en 0):

```
subcadena = cadena.substring(6,5); // subcadena = 'dos'
```

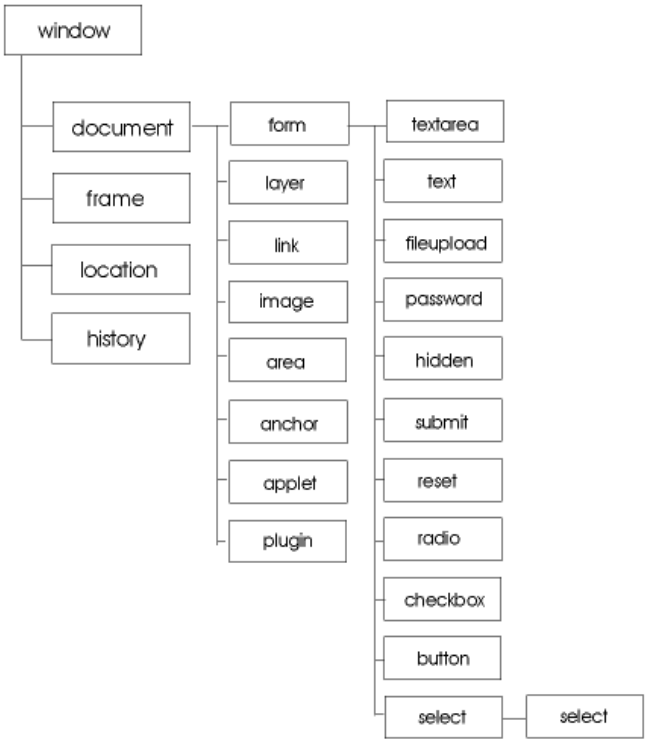
sup() Devuelve una cadena con el formato SUP (superíndice) de HTML

toLowerCase() Devuelve una cadena con los caracteres convertidos a minúsculas

toUpperCase() Devuelve una cadena con los caracteres convertidos a mayúsculas

EXTENSIÓN PARA EL CLIENTE

La **extensión para cliente** permite acceder a los **objetos** que utiliza el **navegador** y al **DOM** (*Document Object Model*), la **jerarquía de objetos** de un documento **HTML**:



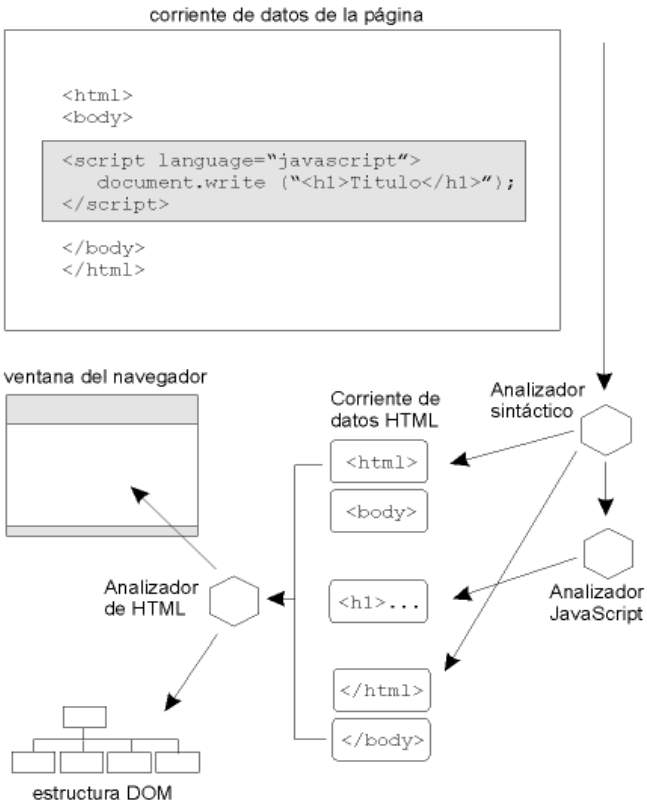
Estos objetos los crea el motor **JavaScript** del navegador (o al menos están disponibles para dicho motor) a partir de la página HTML abierta por el propio navegador.

El proceso comienza cuando el navegador abre una página HTML (por ejemplo solicitándola a un servidor web o directamente del sistema de ficheros local).

La información contenida en dicha página constituye una corriente de datos con un inicio y un final.

El **analizador sintáctico** (**parser**) del navegador lee esa corriente de datos de forma secuencial, extrayendo (y mostrando en pantalla si es necesario) los distintos elementos HTML. Estos elementos también son convertidos a objetos que formarán parte de la estructura del documento (DOM) y que estarán disponibles para la propia funcionalidad del navegador y para los scripts y objetos (plugins) incluidos en la página.

De hecho, los programas JavaScript del lado del cliente pueden interactuar de dos formas con las páginas de las que forman parte:



1.- Durante la carga del documento, es decir, durante el análisis de la corriente de datos, pueden incorporar a dicha corriente expresiones HTML que serán aceptadas por el parser HTML y convertidas en los elementos necesarios. Habitualmente se utiliza este método para añadir información dinámica (fecha, hora, etc...) o para mostrar información en función de alguna condición (tipo de navegador que utiliza el usuario, resolución, etc...).

El proceso se muestra, de forma aproximada, en la figura de la izquierda:

2.- Una vez cargada la página, el documento HTML está reflejado en la estructura de objetos construida por el navegador. Los programas JavaScript pueden acceder a esta estructura para recoger información o incluso para modificarla de forma dinámica. La interactividad con el usuario se consigue a través del mecanismo de captura y tratamiento de eventos.

El acceso a los distintos objetos y sus propiedades se lleva a cabo recorriendo la jerarquía:

```
<html>
<body>
<form name='form1'>
    <input type='text' name='nombre'>
</form>
...
window.document.form1.nombre.value = "Este es el nombre";
...
```

JERARQUIA DE OBJETOS EN JAVASCRIPT

OBJETO WINDOW

El **objeto WINDOW** representa una ventana o marco del navegador. Es el elemento raíz de la estructura de objetos de cada página HTML.

Tiene definidos los eventos: **onBlur**, **onDragDrop**, **onError**, **onFocus**, **onLoad**, **onMove**, **onResize**, **onUnload**.

Propiedades del objeto WINDOW:

Arrays de elementos contenidos en la ventana actual

frames	Array de marcos (frames) de la ventana actual

Otros atributos

closed	Vale true si esa ventana ha sido cerrada
defaultStatus	Mensaje por defecto que se muestra en la barra de estado de esa ventana
document	Objeto document asociado a la ventana
history	Historial de URLs visitadas
innerHeight	Altura del área de contenido de la ventana (en pixels)
innerWidth	Ancho del área de contenido de la ventana (en pixels)
length	número de marcos en la ventana (equivalente a frames.length)
location	Objeto location asociado a la ventana. Contiene información sobre el URL actual.
locationbar	Barra de dirección (URL) del navegador. Contiene a su vez la propiedad 'visible' que indica si se muestra o no la barra en la ventana actual.
menubar	Barra de menú del navegador. Contiene la propiedad 'visible' que indica si se muestra o no la barra en la ventana actual.
name	Nombre de la ventana
opener	Ventana que abrió la ventana actual a través del método open(). Por ejemplo, para saber el nombre de dicha ventana: nombre=window.opener.name;
outerHeight	Altura de la ventana en pixels
outerWidth	Ancho de la ventana en pixels
pageXOffset	Posición horizontal (en pixels) de la página con respecto a la ventana
pageYOffset	Posición vertical (en pixels) de la página con respecto a la ventana
parent	Representa el nombre de la ventana o marco que contiene a la ventana actual
personalbar	Barra de directorios del navegador
scrollbars	Barras de desplazamiento del navegador
self	Representa la ventana actual
status	Mensaje de la barra de estado
statusbar	Barra de estado de la ventana
top	Representa la ventana de nivel superior (raíz) en una estructura de marcos
window	Representa la ventana actual (como self)

Resumen de métodos del objeto WINDOW :

Métodos

alert Muestra una ventana de mensajes. Ejemplo:

```
if (numero>10) alert ("Fuera de rango");
```

back Carga la URL anterior de la lista del objeto HISTORY correspondiente a la ventana de nivel superior:

```
window.back();
```

clearInterval Cancela un temporizador iniciado con ***setInterval***

clearTimeout Cancela un temporizador iniciado con ***setTimeout***

close Cierra la ventana especificada (dependiendo de la seguridad implementada en cada navegador puede que sólo funcione con ventanas abiertas con el *método* ***open***):

```
otraventana = window.open("")
otraventana.close ()
```

find Busca una cadena de texto en el contenido de la ventana especificada:

```
find (cadena, mayusculas, siguiente)
```

Si 'mayusculas' es true, se lleva a cabo una búsqueda que tiene en cuenta mayúsculas y minúsculas. Si 'siguiente' es **true**, se busca la siguiente coincidencia. Si no se especifica 'cadena', el navegador lanza la ventana de búsqueda. La función devuelve **true** si se encontró la cadena.

confirm Muestra una ventana de confirmación con los botones **OK** y **CANCEL**. Devuelve **true** si el usuario pulsa **OK** y **false** en caso contrario.

focus Asigna el foco a un objeto

forward Carga la siguiente URL de la lista del objeto **HISTORY**

handleEvent Llama a la función encargada de manejar el evento especificado

home Carga la URL definida como **'HOME'** en el navegador

moveTo Mueve la ventana a las coordenadas especificadas:

```
window.moveTo(25,10);
```

open Abre una nueva ventana del navegador [*]

print Imprime el contenido de la ventana

resizeBy Redimensiona la ventana moviendo la esquina inferior derecha a las coordenadas especificadas:

```
window.moveTo (100,100);
// redimensionar a 400x400 pixels
window.resizeBy (500, 500);
```

resizeTo Redimensiona la ventana a los nuevos valores de ancho y alto:

```
// redimensionar a 400x400 pixels
window.resizeTo (400, 400);
```

scrollBy Desplaza el contenido de la ventana la cantidad indicada

scrollTo Desplaza el contenido de la ventana

setInterval Establece un temporizador cíclico que se encarga de evaluar una expresión regularmente [*]

setTimeout Establece un temporizador que evalúa una expresión una vez transcurrido el tiempo asignado [*]

stop Detiene el proceso de carga de la página actual

El **método open** se utiliza para abrir nuevas ventanas del navegador. Los parámetros son:

open (URL, nombre_de_la_ventana, opciones);

Las opciones se especifican mediante una secuencia de pares 'parametro=valor' separados por comas (sin espacios en medio):

Algunas opciones del método *open()*

dependent	Crea una ventana hija de la ventana actual. Esta ventana se cerrará automáticamente cuando lo haga su ventana padre.
Height	Altura de la ventana (en pixels)
Location	Si es 'yes', crea una entrada para las direcciones (URLs) en la nueva ventana.
menubar	Si es 'yes', crea una barra de menú en la nueva ventana.
resizable	Si es 'yes', el usuario podrá redimensionar la ventana.
scrollbars	Si es 'yes', aparecerán barras de desplazamiento en la nueva ventana (cuando el documento supere las dimensiones de la misma).
status	Si es 'yes', la nueva ventana tendrá barra de estado.
titlebar	Si es 'yes', aparecerá la barra de título en la nueva ventana.
toolbar	Si es 'yes', la ventana tendrá barra de herramientas.
width	Ancho de la ventana (en pixels)

Por ejemplo, para abrir una página en una nueva ventana:

```
window.open ("pagina2.html", "ventana2" );
```

Una nueva ventana con barra de menu, pero sin barra de estado, ni entrada para direcciones:

```
window.open ("pagina2.html", "ventana2", "status=no, location=no, menubar=yes");
```

Si no se especifica URL, la ventana se abre sin cargar ninguna página. Si la ventana ya existe, no se borra la página que está cargada en ese momento.

El siguiente ejemplo abre una nueva ventana y construye su contenido de forma dinámica (**on-the-fly**):

```
<html>
<head>
<script language="JavaScript">
    <!—
    function abreNuevaVentana(nombreVentana){
            // abrir una nueva ventana con el nombre pasado como parámetro
            // no se especifica una URL
            miVentana= open("", nombreVentana,"width=500, height=400, status=yes,
                                                    toolbar=yes , menubar=yes");
            // Abre la corriente de datos del documento para escribir
            miVentana.document.open();
            // Crea el documento
            miVentana.document.write("<html><head><title>On-the-fly");
            miVentana.document.write("</title></head><body>");
            miVentana.document.write("<h1>Creando documentos al vuelo</h1>");
            miVentana.document.write("<p> </p>");
            miVentana.document.write("<p>El contenido es din&aacute;mico</p>");
            miVentana.document.write("</body></html>");
            // cierra la corriente de datos del documento
            miVentana.document.close();
    }
    // -->
</script>
</head>
<body>
    <form>
            <input type=button value="On-the-fly" onClick="abreNuevaVentana('nueva')">
    </form>
</body>
</html>
```

Los métodos **setInterval** y **setTimeout** permiten establecer **temporizadores** en la ventana. Por ejemplo, el siguiente código muestra un mensaje transcurridos 10 segundos utilizando el método setTimeout:

```
<html>
<head>
<script language="javascript">
    function temporizador(){
            window.setTimeout ("alert('Mensaje')", 10000);
    }
</script>
</head>
<body bgcolor="#FFFFFF">
    <form method="post" action="">
            Muestra un mensaje dentro de 10 segundos
            <input type="button" name="Button" value="ok" onClick="temporizador();">
    </form>
</body>
</html>
```

Para establecer **temporizadores cíclicos** se utiliza el método setInterval. Por ejemplo, para establecer un aviso cada 5 segundos:

```
window.setInterval ("alert ('Han pasado 5 segundos')", 5000);
```

JERARQUIA DE OBJETOS EN JAVASCRIPT
OBJETO DOCUMENT

El **objeto document** se construye a partir de la marca BODY de HTML. Sobre él pueden actuar los eventos: **onClick**, **onDblClick**, **onKeyDown**, **onKeyPress**, **onKeyUp**, **onMouseDown**, **onMouseUp**.

Resumen de propiedades del objeto document:

Arrays de elementos contenidos en el documento actual

plugins	array de plugins del documento
layers	array que contiene las capas del documento
images	array que contiene las imágenes cargadas en el documento
applets	array que contiene los applets del documento
links	array que contiene los enlaces del documento
forms	array que contiene los formularios definidos en el documento
anchors	array que contiene los puntos de anclaje (anchors) del documento

Propiedades del documento (elemento BODY)

bgColor	Color de fondo del documento (atributo BGCOLOR de HTML)
linkColor	Color de los enlaces (atributo LINK de HTML)
alinkColor	Color de los enlaces activos (atributo ALINK de HTML)
vlinkColor	Color de los enlaces visitados (atributo VLINK de HTML)
fgColor	Color por defecto del texto (atributo TEXT de HTML)
lastModified	Fecha en la que fue modificado el documento por última vez
domain	Nombre del servidor web del que procede el documento
title	Título del documento (contenido de la marca TITLE de HTML)
URL	URL completa del documento actual
referrer	URL del documento a través del cual se cargó el documento actual

Resumen de métodos del objeto document:

Métodos

close	Cierra una corriente de datos abierta por el método open y hace que se muestren todos los elementos
getSelection	Devuelve una cadena con el texto seleccionado por el usuario
handleEvent	Llama a la función encargada de manejar un determinado evento
open	Abre una corriente de datos para escribir en el documento (normalmente con los métodos write o writeln)
write	Escribe texto a la corriente de datos del documento
writeln	Igual que write pero finaliza el texto con un retorno de carro

Ejemplos del **objeto document**:

1.- El siguiente ejemplo abre una nueva corriente de datos en un documento y escribe en ella.

```
function corriente ( ){
        // abre una corriente de datos
        document.open();
        // envía datos a la corriente
        document.write ("La corriente est&aacute; abierta");
        // cierra la corriente de datos y visualiza los elementos
        document.close();
}
...
// cuando se pulsa el botón, el documento actual se sustituye
// por los datos que se envían a la corriente de datos
<input type="button" name="Button" value="Button" onClick="corriente();">
```

2.- El texto escrito por el método write durante la carga de una página será considerado como una expresión HTML y añadido tanto a la pantalla del navegador como a la estructura de objetos:

```
<html>
<body>
<script language="javascript'>
    document.write ("<h1> Titulo </h1">;
    document.write ("<p> Esto es un p&aacute;rrafo </p>");
    ...
</script>
</body>
</html>
```

3.- El siguiente código muestra el nombre (si lo tienen) de todas las imágenes cargadas en el documento:

```
function imagenes (){
      // se almacenan los nombres de las imágenes
      // antes de abrir una nueva corriente de datos
      var imagenes = new Array;
      for (a=0; a<document.images.length; a++){
            imagenes[a]= document.images[a].name;
      }
      // abre una corriente
      document.open();
      for (a=0;a<imagenes.length;a++){
            // muestra el nombre y el índice de cada imagen
            document.write (imagenes[a]+"   "+a+"<br>")
      }
      document.close();
}
```

JERARQUIA DE OBJETOS EN JAVASCRIPT
OBJETO LOCATION

Contiene información sobre la dirección de la página actual (URL actual).

Propiedades del objeto LOCATION

hash Nombre de un anclaje en la URL. Por ejemplo, para la dirección: http://www.epsilon-eridani.com/pag1.htm#tema2

```
anclaje = window.location.hash; // anclaje = '#tema2'
```

host Nombre del servidor, su alias o su dirección IP. Para el ejemplo anterior:

```
servidor = window.location.host;
// servidor = www.epsilon-eridani.com
```

hostname Nombre del servidor junto con el puerto en el que está definido el servicio

href Cadena que representa la dirección completa. Para cargar una página diferente en la ventana actual bastaría con ejecutar el siguiente código:

```
window.location.href = "pagina2.html";
```

pathname Directorios que acompañan a la URL de un recurso. Por ejemplo, para la dirección: http://www.punto.com/dir1/dir2/pag.htm

```
directorios = window.location.pathname;
// directorios = /dir1/dir2/pag.html
```

port Puerto en el que está definido el servicio

protocol Protocolo de comunicación especificado por la URL. Para el ejemplo anterior:

```
protocolo = window.location.protocol;
// protocolo = 'http:'
```

search Parámetros que acompañan a la URL. Por ejemplo, para la dirección: http://www.punto.com/cgi-bin/prog?param1=1&param2=2

```
parametros = window.location.search;
// parametros = '?param1=1&param2=2'
```

Métodos del objeto LOCATION

reload Recarga la página actual

replace Carga la URL especificada y la sustituye por la entrada actual del objeto HISTORY

En el siguiente ejemplo se utiliza un botón para cargar una nueva página en la ventana actual:

```
<html>
<head>
</head>
<body bgcolor="#FFFFFF">
    <form method="post" action="">
        <input type="button" name="Button" value="redireccionar"
                onClick="window.location.href='pagina2.html';">
    </form>
</body>
</html>
```

JERARQUIA DE OBJETOS EN JAVASCRIPT
OBJETOS HISTORY Y NAVIGATOR

Objeto HISTORY

Contiene un array con las URLs de las páginas cargadas por la ventana actual.

Propiedades del objeto HISTORY

current	URL de la entrada actual de HISTORY
length	Número de entradas en el historial
next	URL de la entrada siguiente (con respecto a la entrada actual)
previous	URL de la entrada anterior (con respecto a la entrada actual)

Métodos del objeto HISTORY

back	Carga la URL anterior (con respecto a la entrada actual)
forward	Carga la URL siguiente (con respecto a la entrada actual)
go	Carga una URL de la lista. Por ejemplo, para volver a cargar la página una página que se visitó anteriormente: *window.history.go (-1);*

Objeto NAVIGATOR

Permite acceder (modo sólo lectura) a algunas de las características del navegador.

Propiedades del navegador

appCodeName	Nombre en código o alias del navegador utilizado
appName	Nombre del navegador
appVersion	Versión del navegador
language	Idioma del navegador
mimeTypes	Un array con los tipos MIME (Multipart Internet Mail Extension) que soporta el navegador
platform	Plataforma en la que se ejecuta el navegador (Win32, Win16, Mac68k, MacPPC, Unix...)

Por ejemplo, el siguiente código muestra las características del navegador:

```
<script language="javascript">
    document.write (navigator.appCodeName+"<br>");
    document.write (navigator.appName+"<br>");
    document.write (navigator.appVersion+"<br>");
    document.write (navigator.language+"<br>");
    document.write (navigator.platform+"<br>");
</script>
```

Estas propiedades se pueden utilizar para condicionar el comportamiento del programa JavaScript en función del tipo de navegador y de su versión.

TRATAMIENTO DE EVENTOS EN JAVASCRIPT

Los eventos permiten construir aplicaciones interactivas en el lado del cliente. Los eventos están siempre asociados a una acción sobre un elemento determinado. Estos son los eventos más utilizados:

Evento	Tiene efecto sobre	Ocurre cuando	Manejador
Abort	imágenes	El usuario cancela la carga de una imagen	**onAbort**
Blur	ventanas y elementos de formularios	El usuario abandona la ventana o elemento (pierde el foco)	**onBlur**
Change	text fields, textareas, select lists	El usuario cambia el valor de un elemento	**onChange**
Click	buttons, radio buttons, checkboxes, submit buttons, reset buttons, links	El usuario pulsa sobre el elemento o el enlace	**onClick**
Error	images, windows	La carga de la ventana o la imagen causa un error	**onError**
Focus	windows and all form elements	El usuario pasa el control (foco) a un elemento determinado	**onFocus**
KeyDown	documents, images, links, text areas	El usuario pulsa una tecla	**onKeyDown**
KeyPress	documents, images, links, text areas	El usuario da una pulsación con una tecla o la mantiene pulsada	**onKeyPress**
KeyUp	documents, images, links, text areas	El usuario deja de pulsar una tecla	**onKeyUp**
Load	document body	El usuario carga una página en el navegador	**onLoad**
MouseDown	documents, buttons, links	El usuario pulsa una tecla del ratón	**onMouseDown**
MouseMove		El usuario mueve el ratón	**onMouseMove**
MouseOut	areas, links	El usuario mueve el cursor fuera del elemento	**onMouseOut**
MouseOver	links	El usuario mueve el cursor dentro del elemento	**onMouseOver**
MouseUp	documents, buttons, links	El usuario deja de pulsar una tecla del ratón	**onMouseUp**
Move	windows	El usuario (o algún script) mueve una ventana	**onMove**
Reset	forms	El usuario pulsa el botón de reset	**onReset**
Resize	windows	El usuario (o algún script) redimensiona la ventana del navegador	**onResize**
Select	text fields, textareas	El usuario selecciona el contenido de los elementos del formulario	**onSelect**
Submit	forms	El usuario pulsa el botón para enviar un formulario	**onSubmit**
Unload	document body	El usuario sale de la página (carga otra página o cierra esa ventana del navegador)	**onUnload**

Su utilización es muy sencilla. Para indicar a **JavaScript** que se desea hacer el tratamiento de un determinado evento, se coloca el manejador correspondiente en la marca HTML del elemento:

```
<marca_HTML onEvento=expresión_JavaScript>
```

Por ejemplo, podemos hacer que aparezca un mensaje cada vez que el usuario pulsa un botón:

```
...
<input type='button' name='boton1' value='pulsa'
        onClick = 'alert("Se ha pulsado"); return true;' >
...
```

Lo más habitual es colocar una llamada a una función que se utilizará como manejador propiamente dicho del evento:

```
function manejador( ){
// funcionalidad del manejador
    ....
}
...
<input type='button' name='boton1' value='pulsa' onClick = 'manejador()' >
...
```